(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **2000112742 A**

(43) Date of publication of application: **21.04.00**

(51) Int. Cl **G06F 9/06**

(21) Application number: **10285139**

(22) Date of filing: **07.10.98**

(71) Applicant: **FUJITSU LTD**

(72) Inventor: **KONDO TATSUO**
UEHARA TADAHIRO
NAKAYAMA HIROKO
YAMAMOTO RIEKO
OHASHI KYOKO
YOSHIDA HIROYUKI

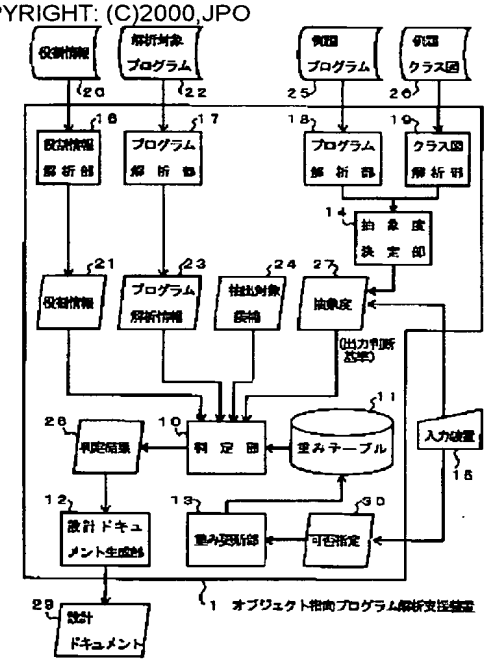
(54) **OBJECT-ORIENTED PROGRAM ANALYSIS SUPPORTING DEVICE AND ITS PROGRAM RECORDING MEDIUM**

COPYRIGHT: (C)2000,JPO

(57) Abstract:

PROBLEM TO BE SOLVED: To improve the precision of judgement by automatically judge whether to extract information which should be reflected to a designing document in an object oriented program analysis supporting device analyzing an object- oriented program to support generation of the designing document.

SOLUTION: A judging part 10 evaluates the importance of the class/member candidate of an extracting object from program analysis information 23 obtained from a program analyzing part 18 and role information 21 obtained from a role information analyzing part 16 to select a candidate to extract according to inputted abstraction degree 27. A designing document generation part 12 generates the designing document 29 such as a class drawing based on a result selected by this part 10 and outputs it. Thus, the document 29 of an abstraction degree according to a purpose is generated.



THIS PAGE BLANK (USPTO)

JP-A-2000-112742

Object-oriented Program Analysis Supporting Device and Program Storage Medium thereof

[0064]

Next, the process flow of each unit of the present invention is described with reference to Figs. 14 through 17. Fig. 14 is a flowchart showing the process of a determination unit 10. Firstly, the determination unit 10 obtains extraction candidates 24 (step S1). Then, the unit 10 checks whether a criterion (degree of abstract) is given from the outside (step S2). If the criterion is not given from the outside, the unit 10 uses a standard criterion (step S3). If the criterion is given from the outside, the unit 10 obtains the criterion (step S4). Furthermore, the unit 10 checks whether there is a weight table 11 (step S3). If there is not the weight table 11, the unit 10 sets all weights (weighted values) equal (step S6). If there is the weight table 11, the unit 10 obtains the weight table 11 (step S7).

[0065]

Then, the unit 10 obtains all criteria (step S8), and extracts one of the extraction candidates 24 (step S9). Then, the unit 10 extracts one criterion of degree

THIS PAGE BLANK (USPTO)

of importance (step S10), computes the respective degree of importance of all the candidates and totals the respective degree of importance with weights (step S11). Furthermore, the unit 10 checks whether each candidate has role information 21 (step S12). Only when a candidate has role information 21, the unit 10 computes the degree of importance of the role information 21 and totals the respective degree of importance of the role information of all the candidates (step S13). The processes in steps 10 through 13 are repeated until the computation of all the criteria are completed (step S14), and the unit 10 determines whether each extraction candidate passes or fails by comparing its computed result with the criteria. In other words, the unit 10 determines whether each of the currently focused extraction candidates 24 should be outputted (step S15). The unit 10 repeats the processes in steps S9 through S15 until the processes of all the extracted candidates 24 are completed (step S16), and outputs a determined result (step S17).

[0066]

Fig. 15 is a flowchart showing the respective processes of program analyzing unit 17 and 18. The program analyzing unit 17 extracts classes from an analysis target program 22 (step S21), and extracts the inheritance relationship between classes for each class

THIS PAGE BLANK (USPTO)

(step S22). Furthermore, the unit 10 extracts members for each class (step S23), and extracts both visibility and virtuality for all the members (step S24).

[0067]

The program analyzing unit 18 also performs the same process for an example program 25. These program analyzing processes are prior arts. Fig. 16 is a flowchart showing the process of a role information analyzing unit 16.

[0068]

The role information analyzing unit 16 extracts design pattern names from role information 20 (step S31), extracts classes constituting a pattern (step S32) and extracts both the general and inheritance relationships between classes (step S33). Then, the unit 16 extracts members which belong to each class (step S34) and extracts the meanings of each class and member (step S35). Then, the unit 16 inputs the result to the determining unit 10 as role information 21.

[0069]

Fig. 17 is a flowchart showing the process of a class diagram analyzing unit 19. The class diagram analyzing unit 19 extracts classes from an example class diagram 26 (step S41) and extracts the inheritance relationship between classes for each class (step S42).

THIS PAGE BLANK (USPTO)

Then, the unit 19 extracts members for each class (step S43), and extracts both visibility and virtuality for all the members (step S44). In this class diagram analyzing process, a prior art can be used.

[0070]

By collating the result analyzed by the program analyzing unit 18 with the result analyzed by the class diagram analyzing unit 19, it can be determined that what degree of abstract the structure of the example program 25 is expressed in the example class diagram 26. An abstract degree determining unit 14 performs such a process. Since the process of generating a design document, such as a class diagram and the like from a real object-oriented program is already known, its detailed description is omitted here. The major difference between the process of the design document generating unit 12 of the present invention and the conventional design document generating process is that in the conventional process, its target output is only full information obtained from a program or specific predetermined information, while in the present invention, information to be outputted is restricted based on a determined result 28 according to a given degree of abstract 27, and a design document 29 is generated in a necessary concept level.

THIS PAGE BLANK (USPTO)

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－112742
（P2000－112742A）
(43)公開日　平成12年４月21日(2000.4.21)

(51)Int.Cl.⁷	識別記号	F I	テーマコード（参考）
Ｇ０６Ｆ　9/06	５３０	Ｇ０６Ｆ　9/06	５３０Ｕ　５Ｂ０７６
			５３０Ｐ
	５４０		５４０Ｕ

審査請求　未請求　請求項の数５　ＯＬ　（全 12 頁）

(21)出願番号　特願平10－285139

(22)出願日　平成10年10月７日(1998.10.7)

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　近藤　竜生
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

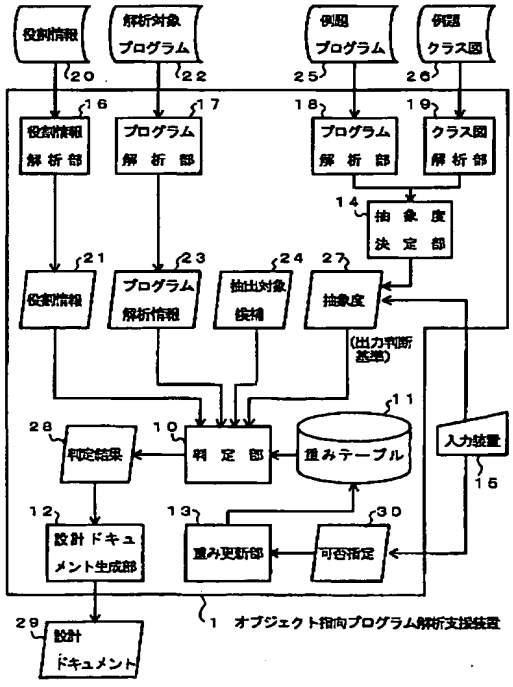
(72)発明者　上原　忠弘
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(74)代理人　100087848
弁理士　小笠原　吉義　（外２名）

最終頁に続く

(54)【発明の名称】　オブジェクト指向プログラム解析支援装置およびそのプログラム記録媒体

(57)【要約】
【課題】　オブジェクト指向プログラムを解析して設計ドキュメントの生成を支援するオブジェクト指向プログラム解析支援装置に関し，設計ドキュメントに反映すべき情報を抽出するか否かの判断を自動的に行い，判断の精度を向上させる。
【解決手段】　判定部10は，プログラム解析部27から得たプログラム解析情報23または役割情報解析部16から得た役割情報21から，抽出対象のクラス／メンバ候補の重要度を評価し，入力した抽象度27に応じて抽出する候補を選択する。設計ドキュメント生成部12は，この判定部10が選択した結果に基づいてクラス図等の設計ドキュメント29を生成し出力する。これにより，目的に応じた抽象度の設計ドキュメント29が生成される。



【特許請求の範囲】
【請求項１】　オブジェクト指向プログラムを解析して設計ドキュメントを生成するオブジェクト指向プログラム解析支援装置であって，オブジェクト指向プログラムを構成するクラスおよびメンバの抽出対象候補と，生成すべき設計ドキュメントの抽象度とを入力とし，前記各抽出対象候補の重要度と前記抽象度との比較により，前記各抽出対象候補を設計ドキュメントに記述すべきか否かを判断する判定手段を持つことを特徴とするオブジェクト指向プログラム解析支援装置。
【請求項２】　請求項１に記載するオブジェクト指向プログラム解析支援装置において，前記判定手段は，プログラム解析情報，またはクラスもしくはメンバの役割情報から得られる複数の重要度の各々に重みを付け，それにより前記各抽出対象候補の総合的な重要度を決定することを特徴とするオブジェクト指向プログラム解析支援装置。
【請求項３】　請求項２に記載するオブジェクト指向プログラム解析支援装置において，前記判定手段の結果に対する利用者からの可否の入力を得て，その可否の結果により，前記重要度に対する重み付けを学習する手段を持つことを特徴とするオブジェクト指向プログラム解析支援装置。
【請求項４】　請求項１に記載するオブジェクト指向プログラム解析支援装置において，前記判定手段が用いる抽象度を，例題となるオブジェクト指向プログラムおよびそのクラス図から推測する抽象度決定手段を持つことを特徴とするオブジェクト指向プログラム解析支援装置。
【請求項５】　オブジェクト指向プログラムを解析して設計ドキュメントの生成を支援する装置を計算機によって実現するためのプログラムを記録した記録媒体であって，オブジェクト指向プログラムを構成するクラスおよびメンバの抽出対象候補と，生成すべき設計ドキュメントの抽象度とを入力とし，前記各抽出対象候補の重要度と前記抽象度との比較により，前記各抽出対象候補を設計ドキュメントに記述すべきか否かを判断する処理を，計算機に実行させるプログラムを記録したことを特徴とするオブジェクト指向プログラム解析支援プログラム記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は，オブジェクト指向プログラムの解析をコンピュータを利用して支援するオブジェクト指向プログラム解析支援装置およびそのプログラム記録媒体に関する。
【０００２】
【従来の技術】オブジェクト指向プログラムの開発はインクリメンタルに行われる。すなわち，最初にプログラムの核となる部分のみを分析・設計し，それに基づいて実装を行い（第１フェーズ），プログラムの核となる部分の動作確認の後，プログラム拡張部分の分析・設計（第２フェーズ）へと進めていく。
【０００３】このとき，第１フェーズの実装工程において，当初の設計にはなかった変更が行われることがある。このような場合には，第２フェーズの開始にあたって，第１フェーズの設計に実装時の変更を反映したものを第２フェーズ設計の基礎とする必要がある。そうでなければ，設計と実装とが食い違ってしまい，実際に作成されたプログラムは設計と異なってしまうという問題が発生する。
【０００４】また，プログラムは，実装される機種やＯＳ等にかかわるような，クラス図等の設計ドキュメントよりもはるかに大量の情報を含む。したがって，第１フェーズのプログラム作成時点での変更を第２フェーズの設計ドキュメントに反映させる際に，プログラムから抽出した情報が，設計ドキュメントに反映されるべき本質的なものであるか，いわば実装詳細のような設計ドキュメントには反映すべきでないものであるかを判断する必要がある。
【０００５】従来，オブジェクト指向プログラムの開発を支援するために，クラス図やメッセージのシーケンス図からプログラムを可能な範囲で自動生成する技術が知られている。一方，これとは逆に，オブジェクト指向プログラムの解析を容易にするために，既存のオブジェクト指向プログラムからクラスやクラス相関等の情報を抽出し，ＵＭＬ（Unified Modeling Language）によるクラス図，シーケンス図等の設定ドキュメントを生成し，出力する方法も考えられている。
【０００６】
【発明が解決しようとする課題】しかし，この従来のオブジェクト指向プログラムの解析を支援する方法の場合には，プログラムから実装詳細情報を含むすべての情報が抽出されてしまうため，結局，設計ドキュメントに必要な情報であるかどうかの判断は，人間が行わなければならず，その判断には，設計やプログラミングについて高度な理解を要するという問題があった。また，大規模かつ複雑なシステムの開発にあっては，正確な判断を保証することがきわめて困難であるという問題もあった。
【０００７】本発明は，上記問題点の解決を図り，従来は人手で行われていたプログラム変更点の判断を自動化することによって，プログラムの変更点の設計ドキュメントへの反映を容易にし，判断の精度を向上させることを目的とする。
【０００８】
【課題を解決するための手段】本発明は，オブジェクト指向プログラムの開発環境においてプログラムを解析して設計ドキュメントを生成する装置において，オブジェクト指向プログラムを構成するクラスおよびメンバを入力とし，設計ドキュメントで記述すべきか否かを判断す

る手段を持つことを主要な特徴とする。判断基準として，生成するクラス図の抽象度を用いる。ここで，抽象度とは，どの程度まで詳しい情報を出力するかの出力レベルを表すものである。

【０００９】出力に対する抽象度が与えられると，プログラムを解析することにより獲得した抽出対象の重要度に応じて，設計ドキュメントへの出力要素が決定される。この重要度は，プログラム内外で与えられるクラスないしメンバの役割情報からも計算される。複数の重要度の各々に重みを付け，それにより総合的な重要度を決定する。

【００１０】抽象度は，インタラクティブに指定されるが，既に生成済みのクラス図と，それの基となったプログラムとを解析して，自動的に推測させることもできる。すなわち，例題クラス図と例題プログラムとから，それに合うような出力レベルでの抽象度が選択され，その抽象度に基づく出力が行われる。

【００１１】さらに，一旦決定された判断に対して，利用者に可否を指定させ，それを学習して以後の判断に役立てるようにするための手段を持つ。

【００１２】

【発明の実施の形態】図１は，本発明の実施の形態の構成例を示すブロック図である。オブジェクト指向プログラム解析支援装置１の判定部１０は，解析対象のオブジェクト指向プログラムを構成するクラスまたはメンバ等の抽出対象候補２４と，出力する設計ドキュメント２９の出力判断基準である抽象度２７とを入力して，解析対象プログラム２２のプログラム解析情報２３から計算した重要度と，プログラムのコメントその他から取得する役割情報２１から計算した重要度とをもとに，その抽出対象候補２４を抽出するかどうかを判断し，その判定結果２８を出力する手段である。判定部１０の入力として，永続的記憶に蓄えられている重みテーブル１１が用いられる。重みテーブル１１は，複数の重要度から総合的な重要度を決めるための各重要度に対する重み付けの値を保持する。

【００１３】設計ドキュメント生成部１２は，判定部１０による判定結果２８に基づいて，必要な抽象度２７で設計ドキュメント２９を生成し出力する。重み更新部１３は，入力装置１５を介して，出力された設計ドキュメント２９の結果に対する利用者の可否指定３０の情報を入力して，利用者が満足できる抽象度による設計ドキュメント２９の出力がなされるように，必要に応じて重みテーブル１１に記憶する重み付けを更新する。この重み付けの更新によって，重みが適当な値になるように学習される。重みテーブル１１が更新されると，判定部１０は，再度，抽出対象候補２４の重要度を計算して，抽出対象候補２４を選別する処理を行い，その結果を判定結果２８として出力する。

【００１４】抽象度２７は，入力装置１５からインタラクティブに入力することもできるが，例題プログラム２５および例題クラス図２６から推測して決定することもできる。抽象度決定部１４は，入力装置１５から抽象度を取得する代わりに，例題プログラム２５および例題クラス図２６を，それぞれプログラム解析部１８とクラス図解析部１９によって解析し，その解析情報から抽象度２７を推測する手段である。

【００１５】役割情報解析部１６は，別ファイル形式の役割情報２０または解析対象プログラム中のコメント等から抽出した役割情報を解析して，各クラスおよびメンバの意味などの役割情報２１を判定部１０に渡す。プログラム解析部１７は，解析対象プログラム２２を解析して，解析結果のプログラム解析情報２３を判定部１０に渡す。プログラム解析部１８は，例題プログラム２５を解析し，クラス図解析部１９は，例題クラス図２６を解析し，それぞれの解析結果を抽象度決定部１４に渡す。プログラム解析部１７，１８，クラス図解析部１９は，それぞれ電子文書の形式である解析対象プログラム２２，例題プログラム２５，例題クラス図２６を解析するが，これらの手段は，公知技術により実現することができる。

【００１６】図１に示す各処理手段を計算機によって実現するためのプログラムは，計算機が読み取り可能な可搬媒体メモリ，半導体メモリ，ハードディスクなどの適当な記録媒体に格納することができる。

【００１７】以下，図２に示すようなオブジェクト指向プログラムを解析対象とする場合の処理について，本発明の実施の形態を詳しく説明する。なお，以下の説明において，クラス名，データメンバ名，メンバ関数名の表記は次のようにしている。

【００１８】・クラス名

先頭のみ大文字の単語を直接結合したもの（例：Ｄｉｒｅｃｔｏｒｙ，ＡｂｓｔｒａｃｔＦｉｌｅ）。

【００１９】・データメンバ名

小文字のみからなる単語をアンダースコア（＿）によって結合し，最後に括弧を伴わないもの（例：ｎａｍｅ，ｐｉｐｅ＿ｎａｍｅ）。

【００２０】・メンバ関数名

小文字のみからなる単語をアンダースコア（＿）によって結合し，最後に括弧を伴うもの（例：ｏｐｅｎ（），ｇｅｔ＿ｎａｍｅ（））。

【００２１】〔１〕抽出対象候補

図３は，抽出対象候補およびプログラム解析情報の例を示す図である。抽出対象候補は，オブジェクト指向プログラムを構成するクラスまたはメンバである。図３（Ａ）に示すように，図２の解析対象プログラムについてのクラスの候補は，ＡｂｓｔｒａｃｔＦｉｌｅ，Ｆｉｌｅ，Ｄｉｒｅｃｔｏｒｙ，Ｓｏｃｋｅｔ，Ｐｉｐｅであり，図３（Ｂ）に示すように，メンバの候補は，ｎａｍｅ，ｐｅｒｍｉｓｓｉｏｎ，ｏｐｅｎ（），ｃｌｏｓ

ｅ（），ｃｒｅａｔｅ（），ｄｅｌｅｔｅ（），ｇｅｔ＿ｎａｍｅ（），ｇｅｔ＿ｐｅｒｍｉｓｓｉｏｎ（），ｇｅｔ＿ａｌｌ＿ｆｉｌｅｓ（），ｂｉｎｄ＿ｐｏｒｔ（），ｐｉｐｅ＿ｎａｍｅである。

【００２２】これらの抽出対象候補は，解析対象プログラムからプログラム解析部１７によってプログラム解析情報の一部として取得してもよく，また，別の入力手段により直接取得してもよい。

【００２３】〔２〕プログラム解析情報

① クラスのプログラム解析情報

クラスのプログラム解析情報として，図３（Ａ）に示すように，例えば，各クラスのメンバ数，他クラスとの関連数，クラスの継承の深さ等を用いる。

【００２４】メンバ数として，メンバ数が多いクラスは構造が複雑であり，より重要であると考えられることから，クラスにはメンバ数に応じた重要度を与える。また，他クラスとの関連数では，他のクラスと密接な関係を持つクラスは重要であると考えられることから，例えば，他のクラスを型として持つデータメンバ数が多いクラスに高い重要度を与える。

【００２５】ただし，ＦａｃａｄｅパターンやＡｂｓｔｒａｃｔＦａｃｔｏｒｙパターンに現れるクラスのように，他のクラスとの関連がきわめて疎であるにもかかわらず，設計段階で記述すべき重要なクラスもある。これらのクラスについては，後述する役割情報を用いて重要度を与える。

【００２６】なお，継承関係の数については，後述するように別項とするため，ここで対象とするのは集約（ａｇｇｒｅｇａｔｉｏｎ）を含む関連（ａｓｓｏｃｉａｔｉｏｎ）のみである。

【００２７】また，クラスの継承の深さでは，継承の深いクラスは，一般的に，より多くのメンバを含み，実装に関する詳細情報の記述という側面を持つ傾向が強い。したがって，深い継承関係が存在する場合には，上位（基底クラス）に出現するクラスほど設計の本質に近いと考えられるため，上位に出現するクラスに対して高い重要度を与える。

【００２８】② メンバのプログラム解析情報

メンバのプログラム解析情報として，各メンバの公開範囲（ｐｕｂｌｉｃ／ｐｒｉｖａｔｅの種別），メンバの抽象性，メンバ関数の複雑さや大きさ等を用いる。図３（Ｂ）は，メンバのプログラム解析情報の例を示す。

【００２９】メンバの公開範囲では，ｐｕｂｌｉｃなメンバ関数は，オブジェクトの振舞いのインタフェースを決定する。したがって，設計ドキュメントで仕様として記述されるべきことが多い。そのため，高い重要度を与える。一方，ｐｒｉｖａｔｅなメンバ関数は，他のメンバ関数が巨大になった場合に分割して処理の見通しをよくするために実装時に加えられることが多いため，これらについては低い重要度を与える。データメンバは，特殊な事情がない限りｐｒｉｖａｔｅであるので，この基準については考えない。

【００３０】また，メンバの抽象性では，類似した機能を持つ複数のクラスでアクセスインタフェースを統一するため，抽象メンバ関数を持つクラスを設け，それを継承するという設計技法がしばしば用いられる。このような場合に，抽象メンバ関数を変更すると他のクラスや関数への影響が非常に大きい。したがって，抽象メンバ関数は，ほとんどの場合に設計ドキュメントに記述されるべき重要性を持つ。

【００３１】メンバ関数の複雑さおよび大きさについて，コンストラクタやデストラクタの処理は，メンバの初期設定や確保したメモリの解放であり処理は単純であると考えられる。また，データメンバを参照したり，値を設定したりするだけの関数（アクセス関数）も，単純な処理しか行わない。これら以外の，他のオブジェクトに問い合わせを行うメンバ関数や自身が持つ複数のデータメンバを関連付けて参照ないし更新するメンバ関数の処理は，複雑であるため高い重要度を与える。

【００３２】また，非効率的なプログラミングを行っていない限り，複雑度とメンバ関数の大きさは正の相関関係を持つ。すなわち，複雑度の代わりにメンバ関数の行数を用いることも可能である。図３（Ｂ）に示すように，本例では，メンバ関数の複雑度を表すものとして，メンバ関数の行数を用いている。なお，行数が少なくても他のオブジェクトに対するアクセス回数が多いものは，処理が複雑であると考えられるので，このようなメンバ関数にも高い重要度を与える。

【００３３】以上はメンバ関数の場合であり，データメンバの場合には，更新や参照の頻度，前後に参照される他のデータメンバの調査など，手続き解析処理を用いて重要度を決定する。処理の詳細はきわめて煩雑であるが，プログラムスライシング（ｐｒｏｇｒａｍ　ｓｌｉｃｉｎｇ）という既存の技術を用いて行うことができるので，ここでのこれ以上の詳しい説明は省略する。

【００３４】〔３〕役割情報

本例では，役割情報としてデザインパターンの構造情報を用いる。デザインパターンとは，ガンマ（E.Gamma ）らが著書「Design Patterns:Elements of Reusable Object-Oriented Software 」でソフトウェア開発への適用を提示した，信頼性の高いシステムを効率的に開発するための概念であり，ソフトウェア設計において，高い頻度で現れる一群のクラスの集まり（パターン）を系統立てて収集し，これらのパターンを組み合わせることで設計の効率化を図るというものである。

【００３５】デザインパターンを構成するクラスのうちのいくつかは，パターン全体のインタフェースを決定するために重要であり，設計ドキュメントで記述されるべき性質のものである。しかし，設計時にはパターンが指定され，それがプログラムとして生成されるが，どのパ

ターンのどの部分としてそのクラスが設計されたのかという情報は，一般にはプログラムの構成要素としては残らない。

【００３６】本例では，デザインパターンに基づく設計ツールにより，パターン名およびクラスのパターン内の位置付け（クラスの役割）をプログラム中のコメントないし別ファイルの形式で残すものとし，これを役割情報として利用する。

【００３７】一例として，デザインパターンの中でも基本的なＣｏｍｐｏｓｉｔｅパターンについて説明する。図４は，Ｃｏｍｐｏｓｉｔｅパターンの例を示す図である。

【００３８】Ｃｏｍｐｏｓｉｔｅパターンは，従来のプログラミング技法で言われるところの木構造を用いるときに現れるパターンであり，Ｃｌｉｅｎｔ，Ｃｏｍｐｏｎｅｎｔ，Ｌｅａｆ，Ｃｏｍｐｏｓｉｔｅの４個のクラスを持つ。

【００３９】Ｃｌｉｅｎｔクラスは，１つの木構造に対して，なんらかの要求を与えるものを示す。Ｃｏｍｐｏｎｅｎｔクラスは，木構造の１つの節（葉もしくは部分木）を表し，これはＣｏｍｐｏｎｅｎｔがＬｅａｆとＣｏｍｐｏｓｉｔｅを派生クラスとして持つということで表現される。Ｌｅａｆクラスは，木構造における葉を表し，節のうち，先に伸びる部分木を持たないものとして定義される。Ｃｏｍｐｏｓｉｔｅクラスは，先にはいくつかの節がついている部分木を表し，これはＣｏｍｐｏｓｉｔｅが任意個のＣｏｍｐｏｎｅｎｔを持つということで表現される。

【００４０】Ｃｏｍｐｏｓｉｔｅパターンにおいて，Ｃｏｍｐｏｎｅｎｔ，Ｌｅａｆ，Ｃｏｍｐｏｓｉｔｅは，ｏｐｅｒａｔｉｏｎ（開発者定義の任意の処理），ａｄｄ（節の追加），ｒｅｍｏｖｅ（節の削除），ｇｅｔ（節の検索）の４つの要求を処理できる。

【００４１】図５は，役割情報の例を示す図である。図２に示す解析対象プログラムの中のクラスまたはメンバが，Ｃｏｍｐｏｓｉｔｅパターン内でどのクラス／メンバに相当するかをパターン内の位置付けとして抽出し，この位置付けにより重要度を与える。図５の例では，クラスについて，Ｃｏｍｐｏｎｅｎｔクラスであれば重要度１００，Ｃｏｍｐｏｓｉｔｅクラスであれば重要度８０，Ｌｅａｆクラスであれば重要度７０としている。また，メンバについて，Ｃｏｍｐｏｎｅｎｔ．ｏｐｅｒａｔｉｏｎ（）であれば重要度１００，Ｃｏｍｐｏｎｅｎｔ．ａｄｄ（）またはＣｏｍｐｏｎｅｎｔ．ｒｅｍｏｖｅ（）であれば重要度７０としている。

【００４２】〔４〕判定部

①　クラスの重要度評価方法

図６は，クラスの重要度評価方法および重みテーブルの初期値の例を示す図である。

【００４３】判定部１０は，クラスの場合には，プログラム解析情報のメンバ数，他クラスとの関連数，継承の深さの各項目の重要度，メンバ重要度および役割情報による重要度から，そのクラスの総合重要度を計算する。評価は以下のようにして行う。

【００４４】メンバ数については，最もメンバの多いクラスを重要度１００，少ないクラスを重要度０とし，メンバ数から最少を減算したものに比例した重要度を与える。他クラスとの関連数については，最も関連数の多いクラスを重要度１００，少ないクラスを重要度０とし，関連数に比例した重要度を与える。継承の深さについては，最も継承の浅い（深さ０）クラスを重要度１００，最も深いクラスを重要度０とし，深さを最大から減算したものに比例した重要度を与える。

【００４５】メンバ重要度については，重要なメンバを持つクラスは重要であると考えられるため，そのクラスにおけるすべてのメンバについて重要度の平均をとる。役割情報による重要度については，クラスに対応する役割情報が与えられている場合は，その情報の持つ重要度をクラスに与える。

【００４６】クラスの総合重要度として，上記の各項目に重みを掛けたもののうち最大のものを取る。ここでは，１つでも突出した重要度を持つクラスは，重要であるとして，その最大の重要度をそのクラスの総合重要度とする。

【００４７】②　メンバの重要度評価方法

図７は，メンバの重要度評価方法および重みテーブルの初期値の例を示す図である。

【００４８】判定部１０は，抽出対象候補がメンバの場合には，プログラム解析情報の公開範囲，メンバの抽象性，メンバの複雑さと大きさ（メンバ関数の行数）の各項目の重要度，役割情報による重要度から，そのメンバの総合重要度を計算する。評価は以下のようにして行う。

【００４９】公開範囲については，メンバ関数が，ｐｕｂｌｉｃならば重要度５０，ｐｒｉｖａｔｅならば重要度１０とする。抽象性については，抽象メンバ関数ならば重要度７０，そうでなければ重要度２０とする。メンバの複雑さと大きさとして，メンバ関数の行数を用い，メンバ関数の実行文が３０以上ならば重要度１００とし，３０未満ならば実行文数に比例した重要度を与える。これはプログラムメトリクスの研究によると，良く設計されたプログラムにおいては，一つの関数の実行文の数は最大３０前後であると言われていることによる。

【００５０】役割情報による重要度として，メンバに対応する役割情報が与えられている場合は，その情報の持つ重要度をメンバに与える。メンバの総合重要度として，上記の各項目に重みを掛けたもののうち最大のものを取る。ここでは，１つでも突出した重要度を持つメンバは，重要であるとして，その最大の重要度をそのメンバの総合重要度とする。

【００５１】図８は，以上のような評価方法により評価した結果の重要度の例を示す図である。図８（Ａ）はクラスの重要度の例，図８（Ｂ）はメンバの重要度の例を示す。図６および図７に示すような評価方法に基づき，クラスのＡｂｓｔｒａｃｔＦｉｌｅについては，メンバ数，関連数，継承の深さ，役割情報による重要度（役割重要度）の重要度が１００であるので，総合重要度を１００とする。また，Ｆｉｌｅについては，役割重要度として重要度７０が与えられているので，総合重要度を７０とする。

【００５２】判定部１０は，以上のクラスの総合重要度およびメンバの重要度と，与えられた抽象度とから，設計ドキュメント２９として出力すべき候補を抽出して，その判定結果２８を設計ドキュメント生成部１２に通知する。設計ドキュメント生成部１２は，判定結果２８に基づき，必要十分な詳しさで設計ドキュメント２９を生成し出力する。

【００５３】重み更新部１３は，入力装置１５を介して判定部１０の処理結果に対する利用者の可否指定３０の情報を入力する。“可”であれば設計ドキュメント生成部１２が出力した設計ドキュメント２９を最終出力とする。“不可”であれば，判定部１０の出力に対して入力装置１５から変更された内容を，新しい判定結果２８として，設計ドキュメント生成部１２へ出力する。この際に，入力装置１５から加えられた変更内容を学習し，必要に応じて重みテーブル１１に記憶する重み付けを修正する。

【００５４】次に，代表的な設計ドキュメント２９の出力例として，クラス図の出力例を説明する。図９～図１１は，本実施の形態において出力されるクラス図の例を示す図である。

【００５５】図９は，抽象度を２０以下にした場合に出力されるクラス図の例である。図８の重要度の例に示すように，本例では候補の総合重要度の最低が２０であるため，抽象度が２０以下の出力が指定された場合には，すべてのクラスおよびメンバの抽出対象候補が選出されることになる。したがって，この場合に出力されるクラス図は，図２に示す解析対象プログラムのすべてのクラス構成を示す図になる。この出力結果は，従来技術による出力結果と同様になると考えてよい。

【００５６】図１０は，出力の抽象度を３０以上にした場合に出力されるクラス図（実装設計レベル）の例を示す。この場合には，総合重要度が３０以上の候補のみを抽出してクラス図が生成され，出力される。したがって，図９に示すクラス図では出力される重要度２０のＰｉｐｅクラスは，この場合には表現されない。

【００５７】図１１は，出力の抽象度を７０以上にした場合に出力されるクラス図（分析設計レベル）の例を示す。この場合には，総合重要度が７０以上の候補のみを抽出してクラス図が生成され，出力される。したがって，図１０に示すクラス図では出力されるＳｏｃｋｅｔクラスは，重要度が７０以下であるので，この場合には表現されない。

【００５８】このように，利用者は，さまざまな抽象度を与えることにより，適切なレベルのクラス図を得ることが可能となる。

〔５〕抽象度決定部

抽象度は，入力装置１５を介して利用者が指定した値を直接入力してもよく，また，例題プログラム２５および例題クラス図２６を入力として，プログラム解析部１８およびクラス図解析部１９により取得した解析情報から，抽象度決定部１４により決定することもできる。抽象度決定部１４は，具体的にはプログラム解析部１８の解析結果について，判定部１０と同様な重要度の計算を行い，どのレベルの重要度まで選択すれば例題クラス図２６と同レベルのクラスまたはメンバが抽出されるかをチェックして，抽象度を推測する。

【００５９】〔６〕重み更新部

判定部１０および設計ドキュメント生成部１２の処理により，クラス図等の設計ドキュメント２９を生成して利用者に提示した後，重み更新部１３は，入力装置１５を介して利用者の可否指定３０を，その理由とともに入力する。

【００６０】重み更新部１３は，入力した可否指定３０の理由に基づいて重みテーブル１１中の該当する項目を特定し，その重みの値を修正する。例えば，図１１に示すクラス図を提示した後，出力結果に対して「ｐｕｂｌｉｃなメンバ関数はなるべく抽出する」という否定理由が入力された場合には，メンバ関数がｐｕｂｌｉｃである場合の重み付けの値を増加させる。

【００６１】図１２は，重みテーブルの変更の例を示す。この例では，「ｐｕｂｌｉｃなメンバ関数はなるべく抽出する」という否定理由の入力により，メンバの公開範囲の重みを１から１．６に変更している。これにより，公開範囲のｐｕｂｌｉｃに与えられる総合重要度を決定する際の重要度は，５０×１．６＝８０になる。

【００６２】図１３は，図１１に示すクラス図について重み付けの値を修正した後に，出力されたクラス図の例を示している。すなわち，抽象度７０以上の出力が指定された場合，ｐｕｂｌｉｃメンバ関数の重み付けの値が１．６になると，その重要度が５０から８０になるので，重み付けの値の変更により，いままで表現されていなかった，メンバのｇｅｔ＿ｎａｍｅ（），ｇｅｔ＿ｐｅｒｍｉｓｓｉｏｎ（）が表現される。

【００６３】なお，複数の否定理由が与えられた場合の重み付け決定アルゴリズムは，例えばニューラルネットワーク技術として公知の技術を利用できるので，ここではその詳細な説明を省略する。

【００６４】次に，図１４～図１７を参照して本発明の各手段の処理の流れを説明する。図１４は，判定部１０

の処理フローチャートである。判定部１０は，まず抽出対象候補２４を取得する（ステップＳ１）。次に，外部から判断基準（抽象度）が与えられているかどうかを調べ（ステップＳ２），外部から判断基準が与えられていない場合には標準判断基準を使用し（ステップＳ３），外部から判断基準が与えられている場合にはその判断基準を取得する（ステップＳ４）。さらに，重みテーブル１１が存在するかどうかを調べ（ステップＳ５），重みテーブル１１が存在しない場合には全ての重み（重み付けの値）を同一にし（ステップＳ６），重みテーブル１１が存在する場合には重みテーブル１１を取得する（ステップＳ７）。

【００６５】次に，全判定要素を取得し（ステップＳ８），抽出対象候補２４を１つ取り出する（ステップＳ９）。続いて，重要度の判定要素を１つ取り出し（ステップＳ１０），判定要素についての候補の重要度を計算し，重み付きで合算する（ステップＳ１１）。さらに候補に対応する役割情報２１があるかどうかを調べ（ステップＳ１２），役割情報２１がある場合にのみ役割情報２１の重要度を計算して合算する（ステップＳ１３）。全ての判定要素について計算が終わるまで，ステップＳ１０～ステップＳ１３の処理を繰り返し（ステップＳ１４），計算結果を判断基準と比較して合否を決定する。すなわち，現在着目している抽出対象候補２４を出力するか否かについて決定する（ステップＳ１５）。すべての抽出対象候補２４について処理が終了するまで，ステップＳ９～ステップＳ１５の処理を行い（ステップＳ１６），判定結果２８を回答する（ステップＳ１７）。

【００６６】図１５は，プログラム解析部１７，１８の処理フローチャートである。プログラム解析部１７は，解析対象プログラム２２からクラスを抽出し（ステップＳ２１），各クラスについてクラス間の継承関係を抽出し（ステップＳ２２），さらに，各クラスについてメンバを抽出し（ステップＳ２３），すべてのメンバについて可視性と仮想性を抽出する（ステップＳ２４）。

【００６７】なお，プログラム解析部１８も，例題プログラム２５に対して同様の処理を行う。これらのプログラム解析処理は，既存の技術である。図１６は，役割情報解析部１６の処理フローチャートである。

【００６８】役割情報解析部１６は，役割情報２０からデザインパターン名を抽出し（ステップＳ３１），パターンを構成するクラスを抽出し（ステップＳ３２），クラス間の関係および継承関係を抽出し（ステップＳ３３），各クラスに属するメンバを抽出し（ステップＳ３４），各クラスおよびメンバの意味を抽出する（ステップＳ３５）。その結果を役割情報２１として，判定部１０の入力とする。

【００６９】図１７は，クラス図解析部１９の処理フローチャートである。クラス図解析部１９は，例題クラス図２６からクラスを抽出し（ステップＳ４１），各クラスについてクラス間の継承関係を抽出し（ステップＳ４２），各クラスについてメンバを抽出し（ステップＳ４３），すべてのメンバについて可視性と仮想性を抽出する（ステップＳ４４）。このクラス図解析処理では，既存の技術を用いることができる。

【００７０】プログラム解析部１８による解析結果と，クラス図解析部１９による解析結果とを照合すれば，例題クラス図２６が，例題プログラム２５の構造をどの程度の抽象度で表現しているかを判別することができる。抽象度決定部１４は，その処理を行う。また，実際のオブジェクト指向プログラムからクラス図等の設計ドキュメントを生成する処理は，既知であるので，ここでの詳細な説明は省略する。本発明に係る設計ドキュメント生成部１２の処理と，従来の設計ドキュメント生成処理との大きな違いは，従来，プログラムから得られるすべての情報またはあらかじめ定められた特定の情報だけを出力対象としていたのに対し，本発明では，与えられた抽象度２７に応じた判定結果２８に基づいて出力する情報を制限し，必要な概念レベルで設計ドキュメント２９を生成する点である。

【００７１】

【発明の効果】以上説明したように，本発明によれば，プログラム変更点を設計ドキュメントへ反映する場合に，利用者が所望する概念レベルに対応して，設計ドキュメントに反映すべき情報の抽出の判断を自動化することができる。これにより，設計ドキュメントへの反映作業の工数を低減でき，オブジェクト指向ソフトウェアの開発を効率化することが可能になる。特に，従来，設計ドキュメントへの反映では，設計およびプログラミングに対する高度な理解が必要であったのに対し，それが不要となり，複雑なシステムの開発において効果が大きい。

【図面の簡単な説明】

【図１】本発明の実施の形態の構成例を示すブロック図である。

【図２】解析対象プログラムの例を示す図である。

【図３】抽出対象候補およびプログラム解析情報の例を示す図である。

【図４】Ｃｏｍｐｏｓｉｔｅパターンの例を示す図である。

【図５】役割情報の例を示す図である。

【図６】クラスの重要度評価方法および重みテーブルの初期値の例を示す図である。

【図７】メンバの重要度評価方法および重みテーブルの初期値の例を示す図である。

【図８】重要度の例を示す図である。

【図９】出力されるクラス図の例を示す図である。

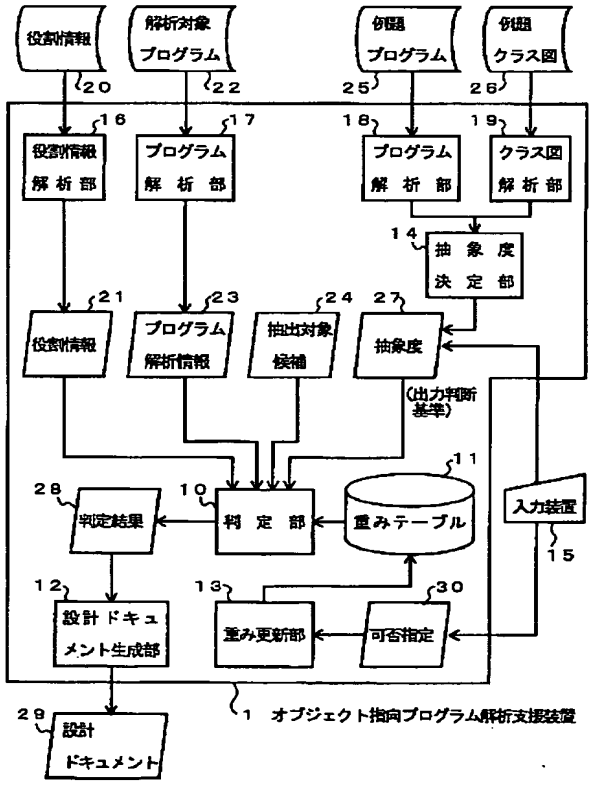
【図１０】出力されるクラス図の例を示す図である。

【図１１】出力されるクラス図の例を示す図である。

【図１２】重みテーブルの変更の例を示す図である

【図１３】図１１に示すクラス図について重み付けの値を修正した後，出力されるクラス図の例を示す図である。
【図１４】判定部の処理フローチャートである。
【図１５】プログラム解析部の処理フローチャートである。
【図１６】役割情報解析部の処理フローチャートである。
【図１７】クラス図解析部の処理フローチャートである。
【符号の説明】
１　オブジェクト指向プログラム解析支援装置
１０　判定部
１１　重みテーブル
１２　設計ドキュメント生成部
１３　重み更新部
１４　抽象度決定部
１５　入力装置
１６　役割情報解析部
１７　プログラム解析部
１８　プログラム解析部
１９　クラス図解析部
２０　役割情報
２１　役割情報
２２　解析対象プログラム
２３　プログラム解析情報
２４　抽出対象候補
２５　例題プログラム
２６　例題クラス図
２７　抽象度
２８　判定結果
２９　設計ドキュメント
３０　可否指定

【図１】



【図２】

解析対象プログラムの例

```
class AbstractFile {
private:
  String name;
  Permission permission;
public:
  virtual void open();
  virtual void close();
  virtual void create();
  virtual void delete();
  String get_name();
  Permission get_permission();
};

class File : public AbstractFile {
};

class Pipe : public File {
private:
  PipeName pipe_name;
};

class Socket : public File {
public:
  void bind_port();
};

class Directory : public AbstractFile {
private:
  vector<AbstractFile> files;
public:
  void get_all_files();
};
```

【図３】

抽出対象候補およびプログラム解析情報の例

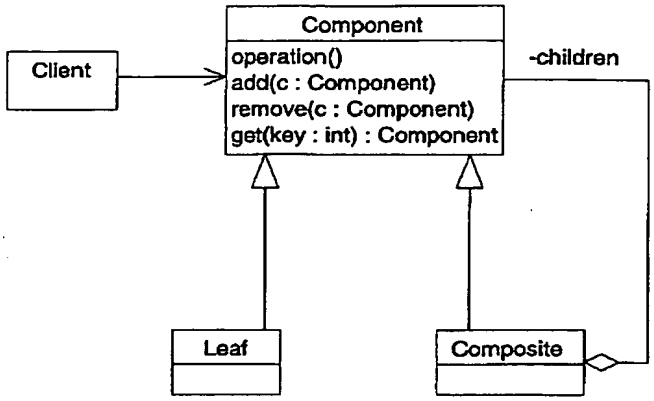
（A）クラスの候補

名　前	メンバ数	他クラスとの関連数	継承の深さ
AbstractFile	8	1	0
File	0	0	1
Directory	1	1	1
Socket	1	0	2
Pipe	1	0	2

（B）メンバの候補

名　前	所属クラス	公開範囲	抽象性	メンバ関数の行数
name	AbstractFile	private	no	—
permission	AbstractFile	private	no	—
open()	AbstractFile	public	yes	5
close()	AbstractFile	public	yes	5
create()	AbstractFile	public	yes	8
delete()	AbstractFile	public	yes	8
get_name()	AbstractFile	public	no	1
get_permission()	AbstractFile	public	no	1
get_all_files()	Directory	public	no	3
bind_port()	Socket	private	no	10
pipe_name	Pipe	private	no	—

【図４】



【図５】

役割情報の例

クラス名ないしメンバ名	パターン名	パターン内の位置付け	位置付けによる重要度
AbstractFile	Composite	Component	100
File	Composite	Leaf	70
Directory	Composite	Composite	80
AbstractFile.open()	Composite	Component.operation()	100
AbstractFile.close()	Composite	Component.operation()	100
AbstractFile.create()	Composite	Component.add()	70
AbstractFile.delete()	Composite	Component.remove()	70

【図６】

クラスの重要度評価方法の例

評価項目	重み初期値	評　価　方　法
メンバ数	1	最もメンバの多いクラスを重要度１００，少ないクラスを重要度０とし，メンバ数から最少を減算したものに比例した重要度を与える
他クラスとの関連数	1	最も関連数の多いクラスを重要度１００，少ないクラスを重要度０とし，関連数に比例した重要度を与える
継承の深さ	1	最も継承の浅い（深さ０）クラスを重要度１００，最も深いクラスを重要度０とし，深さを最大から減算したものに比例した重要度を与える
メンバ関数重要度	1	すべてのメンバについて重要度の平均をとる
役割情報による重要度	1	クラスに対応する役割情報が与えられている場合は，その情報の持つ重要度をクラスに与える
総合重要度	—	上記の各項目に重みを掛けたもののうち最大のものをとる。（１つでも突出した重要度を持つクラスは重要であるとする）

【図７】

メンバの重要度評価方法の例

評価項目	重み初期値	評価方法
公開範囲	1	publicならば重要度５０，privateならば重要度１０とする
抽象性	1	抽象メンバ関数ならば重要度７０，そうでなければ重要度２０とする
複雑さと大きさ	1	実行文が３０以上ならば重要度１００とし，３０未満ならば実行文数に比例した重要度を与える
役割情報による重要度	1	メンバに対応する役割情報が与えられている場合は，その情報の持つ重要度をメンバに与える
総合重要度	―	上記の各項目に重みを掛けたもののうち最大のものをとる。（１つでも突出した重要度を持つメンバは重要であるとする）

【図８】

重要度の例

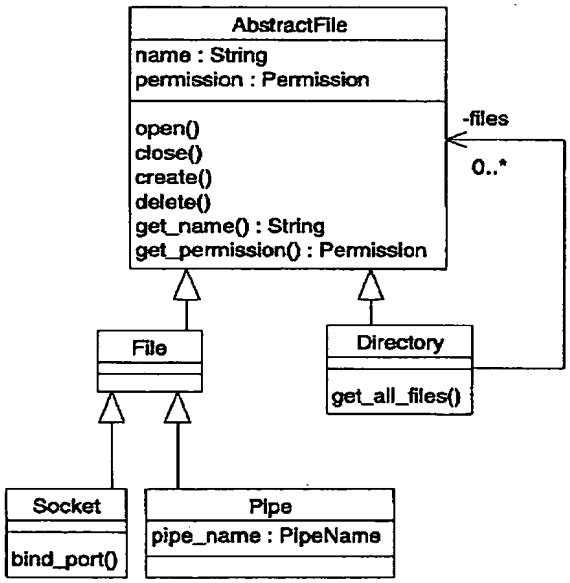
(A) クラスの重要度

名前／評価項目	メンバ数	関連数	承継の深さ	メンバ重要度	役割重要度	総合重要度
AbstractFile	100	100	100	60	100	100
File	0	0	50	0	70	70
Directory	12.5	100	50	50	80	100
Socket	12.5	0	0	33.3	0	33.3
Pipe	12.5	0	0	20	0	20

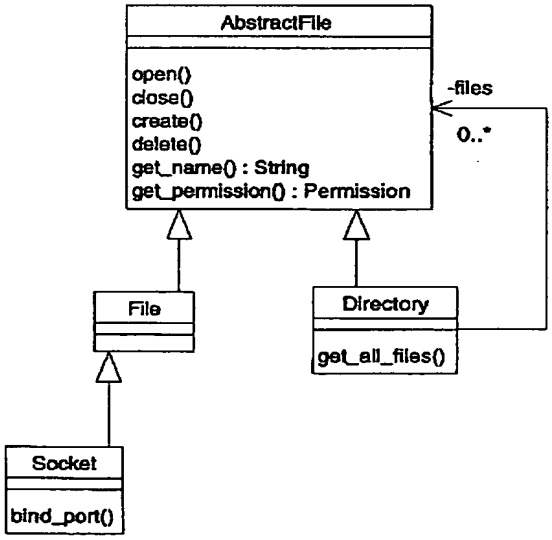
(B) メンバの重要度

名前／評価項目	公開範囲	抽象性	メンバ関数の行数	役割重要度	総合重要度
AbstractFile.name	10	20	―	0	20
AbstractFile.permission	10	20	―	0	20
AbstractFile.open()	50	70	16.7	100	100
AbstractFile.close()	50	70	16.7	100	100
AbstractFile.create()	50	70	26.7	70	70
AbstractFile.delete()	50	70	26.7	70	70
AbstractFile.get_name()	50	20	3.3	0	50
AbstractFile.get_permission()	50	20	3.3	0	50
Directory.get_all_files()	50	20	10.0	0	50
Socket.bind_port()	10	20	33.3	0	33.3
Pipe.pipe_name	10	20	―	0	20

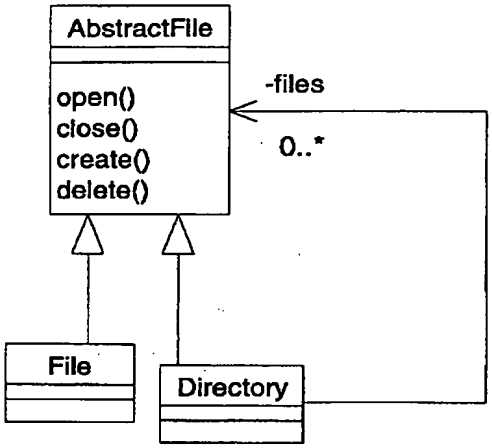
【図９】



【図１０】



【図１１】

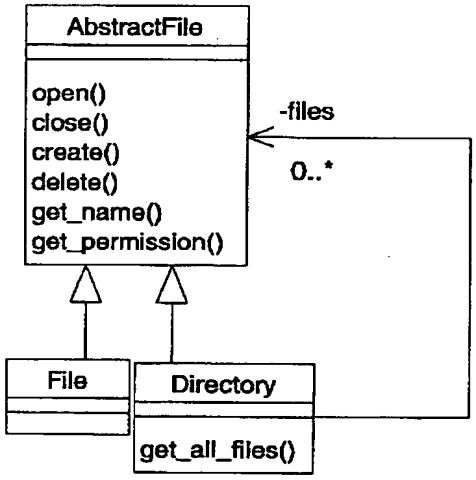


【図１２】

重みテーブルの変更の例

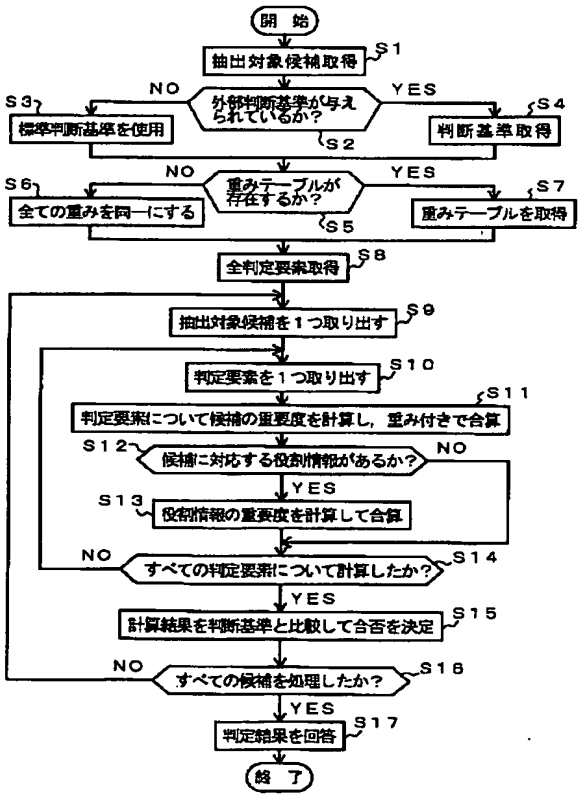
評価項目	重みの初期値	更新後の重み
クラス		
メンバ数	1	1
他クラスとの関連数	1	1
継承の深さ	1	1
メンバ重要度	1	1
役割情報による重要度	1	1
メンバ		
公開範囲	1	1.6
抽象性	1	1
複雑さと大きさ	1	1
役割情報による重要度	1	1

【図１３】



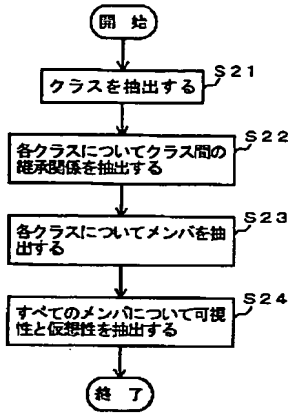
【図１４】

判定部の処理フローチャート



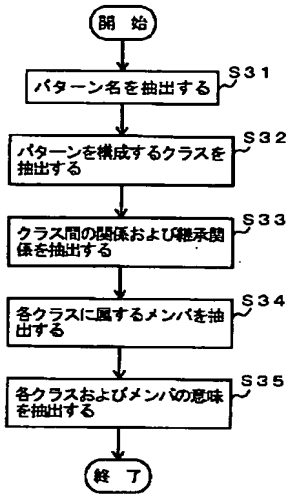
【図１５】

プログラム解析部の処理フローチャート



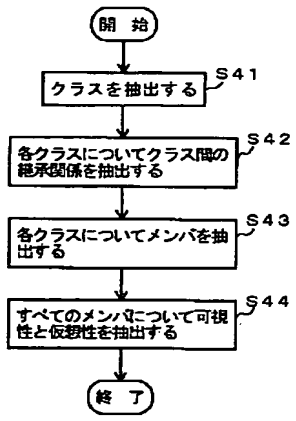
【図１６】

役割情報解析部の処理フローチャート



【図１７】

クラス図解析部の処理フローチャート



フロントページの続き

(72)発明者　中山　裕子
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　山本　里枝子
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　大橋　恭子
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　吉田　裕之
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

Ｆターム(参考)　5B076 DE04